08 | 其它技能考察：见微知著，不可忽略的其它考察点

2021-04-22 四火

技术面试官识人手册 进入课程 >

讲述：四火
时长 17:22 大小 15.91M

你好，我是四火。

关于怎样主导技术面试的问题，在前面几讲里，我们分别对算法和数据结构，以及系统设计这两个最常见的考察路径，做了重点讲解。

而在技术考察方面，还有一些常见的其它路径，比如面向对象的考察，还有针对测试能力的考察等等。

据我观察，近几年来大家对于软件工程师的全面性要求越来越高，早就不再是会写代码、会编辑、会设计系统就可以轻松当面霸了，在这一讲中，我就介绍一些比较常见的其它面试路径。

其它工程技能考察

面向对象考察

严格说起来，多数情况下，这一点应该叫做代码设计能力考察，只不过面向对象是最常见的其中一个方面而已。从技术角度上说，它包含了两个方面：

1.**考察候选人代码层面的面向对象设计能力；**

2.**考察候选人代码层面的综合代码组织的能力。**

先来说说上面的第 1 点。面向对象考察，尤其是对于入职没多久的程序员候选人来说，这是一个非常常见的形式。我们有时讲的代码层面的建模，其实就是通过面向对象的办法，把实际问题使用代码的类、方法等抽象来描述的过程。

既然重点是面向对象，这就意味着往往问题并不夹带复杂的算法逻辑，而面试官的重点也将放在代码设计的层面，能否将实际问题场景的核心抽象成一个又一个的类、方法和属性，安排它们之间的关系，并且应用合适的设计模式，作出合理的解耦，最终落地到代码上。

这种考察方式其实很考验候选人代码层面的设计能力，又和同为代码层面的算法能力有明显区别。我们也确实见到有一些候选人，他们虽然能够攻克复杂的算法，但是在代码设计和组织上却不尽人意。

极客时间

小技巧之引入图示

千言万语不及一幅画，面向对象能力的考察方面，对候选人很有效的一个引导方式，就是给出一个问题的描述，然后要求候选人使用一个 UML 类图，来呈现他的代码设计。

再来说说上面的第 2 点，综合代码组织的能力。如果我们再站高一点看，其实好的代码都不一定非得是面向对象的，即便是过程式的代码，一样可以做好模块化、解耦，因此本质上我们考察的重点是候选人综合的代码组织能力。

另外，这样的面试也需要对编程语言的许多语言特性有准确的认知。比方说，如果面试题目明确指出了要求面向对象的方式来完成实现，那么除了代码设计层面，我们也能看出候选人对于封装、继承、多态等等一些现代编程语言基本特性的理解如何。

无论这一轮面试关注于上面说的哪一点，都要像我们前几讲介绍的那样，专注于具体问题的解决，即和候选人一起使用这些技能和工具解决实际问题，而不是单纯地做设计原则和设计模式理论和概念的问答。

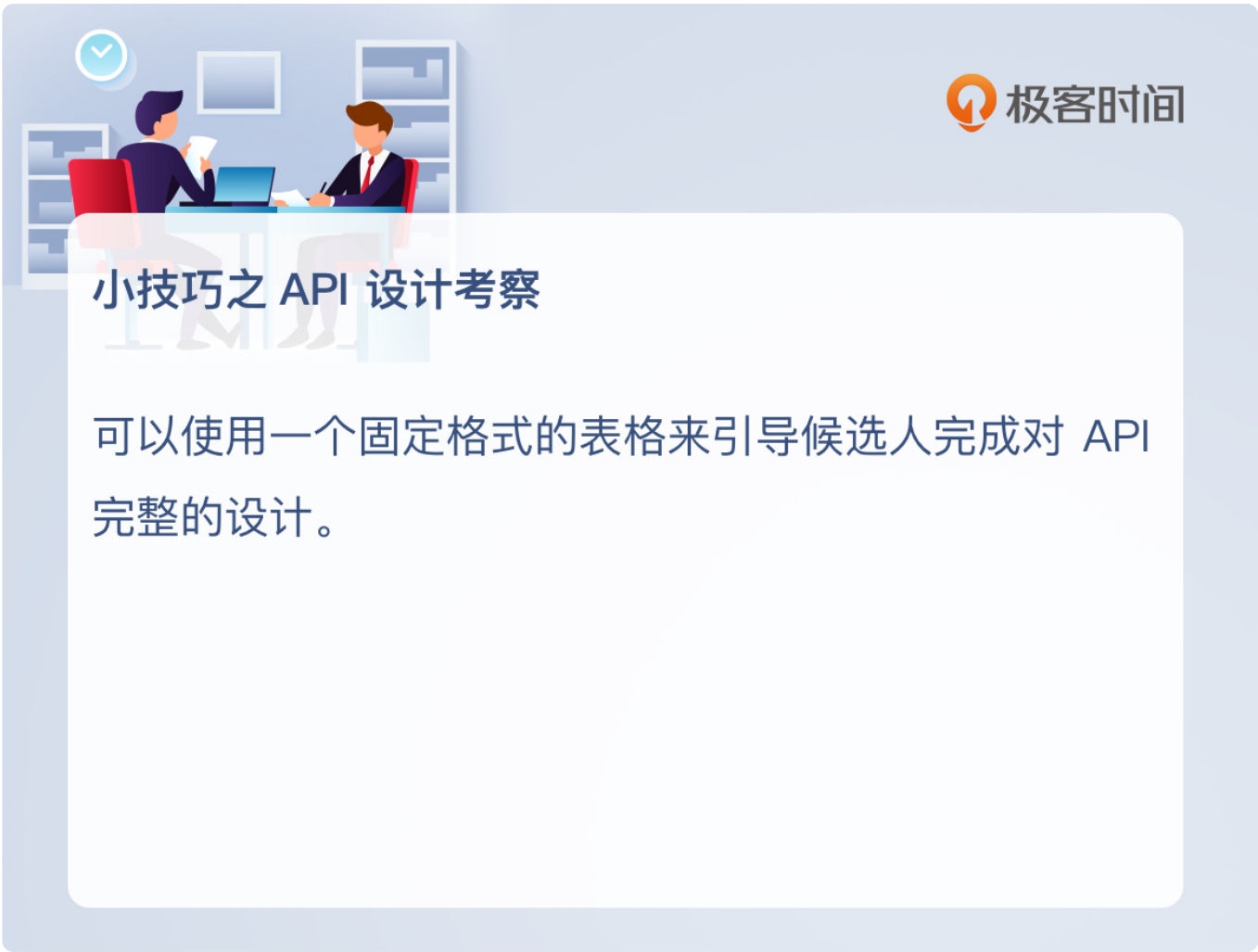
事实上，**面向对象考察中，我观察到最常见的误区，恰恰就是纠结于设计模式。**说到底，代码层面的设计，最终目的是为了代码具备更好的可读性、可重用性和可扩展性。

因此，分层清晰，做到了良好的模块化、做到了合适的解耦，又留下了一定的扩展能力的代码，往往就已经是好代码了，不一定非得扯上某个著名的设计模式；反过来也一样，即便能够准确地说出某个设计模式的定义，它也不能作为对于代码设计能力的有力证明。

API 设计考察

现在对于很多大厂来说，业务都分得很细致，那么每一个团队往往都要维护若干个 service，并且将业务能力通过 API 的形式暴露出来。在这种情况下，API 的调用方就是团队的用户。

因为 API 的设计能力显得举足轻重，所以针对它的考察能够从相当程度上，看出候选人是否在代码接口和客户端与 service 的交互方面有一个平衡的理解。前者是对代码的理解，而后者是对系统的理解。



比方说，还记得第 7 讲中谈到的网约车系统吗？对于其中的 Ride Service，我们就可以引导候选人完成类似如下的 API 接口设计表格：

API 接口设计表

URI	HTTP 方法	入参	返回值	异常	描述

/ride	POST	{
		"userId": xxx,
		"location": {
		"longitude": xxx,
		"latitude": xxx
		},
		"destination": {
		"longitude": xxx,
		"latitude": xxx
		},
		"time": xxx
		}
			{
			"id": xxx
			}
				...	乘客要求乘车从当前 location 到 destination，ride 创建，并返回ride id

...	...				

假如说我们把这个提供 API 的模块当作一个黑盒，当它所有的 API 都已经清楚了，我们对于这个黑盒的功能，以及这个黑盒的用户怎样和它进行交互，这两件事情也就非常明确了。

上面这个 API 的设计，是从 HTTP 协议，或者说是从 REST 服务的角度来描述的，对于不符合这种情况的，或者候选人不熟悉网络协议，我们也可以简单从编程语言代码中方法签名的角度描述，说清楚方法的作用，入参、返回值和数据结构，以及常见的异常种类，其本质是一样的：

复制代码

```
1 Response createRide(Request req);
```

测试能力考察

测试能力的考察，在某些北美的互联网大厂中更为常见。比方说，Google 的工程师面试 loop 中，经常会单独放一轮由测试工程师主导的测试能力考察面试。

你可能会好奇，为什么这些大厂格外重视这点呢？

这里面的初衷是这样的，测试能力是对于一个具备全栈能力的工程师，所必须包含的一项重要素质，而这些企业的大多数负责开发的工程师岗位，都不会再匹配大量的测试工程师

来完成项目，因此需要开发独立完成测试。

说白了，就是自己写的代码要自己测试，自己上线，自己 oncall 和修复问题，完成闭环。

对于测试能力的考察，有两种常见形式。

第一种形式是针对已经完成的代码，做白盒测试。面试官会期望候选人，能够拿着简单的测试用例来走一遍代码。之后也可能会提问，怎样设计测试用例才能将刚写的代码覆盖到，甚至要求写一小段单元测试代码，这是属于白盒测试。

白盒测试一般时间需求比较少，适合结合在每一轮包含编码的面试中，在编码完成后进行拓展。这种形式除了对于测试本身的考察，对于候选人在代码层面的理解也有着很明确的要求。

而第二种形式，则是黑盒测试。对于某一个产品，或者某一个功能特性，从黑盒的角度讨论，怎样才能做好它的测试。黑盒测试主要出现在单独的测试能力考察轮次，这种形式除了对于测试本身的考察，也要求候选人在系统和产品的层面有一定的理解。

举例来说，一个经典的黑盒测试能力考察的面试题是：

> 如果你负责 Google Map 地图的设计，系统给出了从 A 点到 B 的通行路径，你计划做怎样的测试，来保证从 A 到 B 的路径这一结果是合理的？

两种形式，黑盒白盒虽然差异很大，但是很多独立完成测试的工程师所需要的重要素质，比如问题场景的分析能力和用例设计的能力，其实都覆盖到了。

项目与任务管理

我们再来说说项目管理与任务管理方面的考察。在很多大厂的面试中，有一轮经常是由 PM 来负责的，这个 PM 有时候是 Project Manager，更多时候则是 Program Manager，他们会比较关注项目、任务和软件工程的流程方面。

最常见的考察方法是问经历，即直接了当地询问候选人，当前团队中，项目是怎样管理的，产品是怎样上线的，任务是怎样管理的，优先级又是怎样排的。其中很大一部分都可以用询问行为型问题的方法来操作，而这部分我们将在后面重点介绍。

好，上面介绍了一些常见的工程技能的考察类型，但是还有一些其它类型我并没有逐一展开来细数，比如针对产品思维的考察，一些在招聘团队中的产品经理很喜欢这种形式，抛出一个问题，和候选人一起挖掘用户的痛点，从产品角度讨论设计等等。

行为型问题

最后我想介绍一下行为型问题（Behavior Question），这是一种非常流行的非具体技术考察方式。这种方式不光很多重视价值观、领导力的大厂愿意采用，其中很多招聘团队中的 Hiring Manager 更是特别热衷于它。

概念与逻辑

首先，我们必须要弄清楚一个概念，什么是行为型问题？

行为型问题基本上是一类用来观察候选人过去在特定的工作情境下，是怎样解决困难并取得成功的问题。

这里要求候选人所描述的情境，包含的内容非常广泛，可以是与内部同事之间的，可以是与外部客户之间的，可以是关乎项目和任务的，可以是业务决策方面的，也可以是具体技术实现上的。

无论哪一种，**这里面的逻辑是，候选人在过去遇到困难的时候，遵循的逻辑和采取的行为，这相当程度上反映了未来候选人将怎样应对类似的困难。**

如果你还觉得不太清楚的话，可以看看下面这个行为型问题的例子：

> 你能否告诉我一个实例，让我了解你在工作中是怎样说服同事，采纳你的技术决策的？

从这个问题的回答，我们能够看出很多内容，譬如说，对于不同的技术决策，候选人是根据怎样的标准来评估优劣的，候选人是怎样和同事沟通的，等等。通过这样的问答方式，面试官可以看出候选人的许多品质，比如这个例子中，候选人对问题的分析思考能力，和同事的交流沟通能力，以及是否具备 backbone（不轻易动摇）等等。

再比如下面这样的例子：

> 你在工作中是否遇到过不同意你主管（经理）看法的时候？你又是怎样处理的？
> 对于预先订立的项目目标，你有没有遇到过未能按时实现的情况？
> 对于在你做过的项目中的软件设计，有没有事后你觉得自己做错了的？

原则与窍门

好，看完例子，我来总结一下，如果要向候选人提出行为型问题，有这样几个原则和窍门：

第一，问题都是基于“过去”的，或者说，我们希望知道的，都是活生生的，已经在候选人身上发生了什么事情，他抱了怎样的看法，又是怎样应对的。与之相对的是，“假如”型问题，比如：

> 假如你觉得同事的 code change 给系统造成了一个很严重的隐患，但是他又拒绝修改，你该怎么办？

这样的问题当然有它的价值，但它就不属于行为型问题，也不符合前面所说的“过去反映未来”的逻辑来得到考察数据，以帮助我们对候选人做出评估。

为什么这么说？因为所有人是可以“假想”的，但是假想并不能像过去的“事实”一样反映他真的会那样做。

第二，情境要包含冲突，如果可能，最好是一个棘手的冲突。不是所有的行为型问题都要求包含“冲突”的，比方说：

> 介绍一个你自己认为最为成功的项目，并说明为什么你觉得它是成功的？

这个问题本身就不包含明显的冲突，没有冲突就不利于我们知道，在“困难”的情境下，候选人是怎样应对的。当然，这样温和的问题可以进一步发散开去，引出更多尖锐的、包含冲突的问题，那当然是另一回事。

反过来，看看前面我前文中举的一些例子，比如“和主管看法不同”的例子，就隐含了一个和主管观点不一致的冲突。在面试双方能够顺利沟通的前提下，挖掘一些对于尖锐冲突场景应对的事例，可以很好地帮助我们了解候选人。

第三，追问并达到一定深度。这一点和前面几讲中提到的第 7 讲中怎样询问项目一样，我们当然不是希望事无巨细，但有深度的挖掘能在一定程度上帮助确认这些都是发生过的事实，而非随口应付而编造的答案。

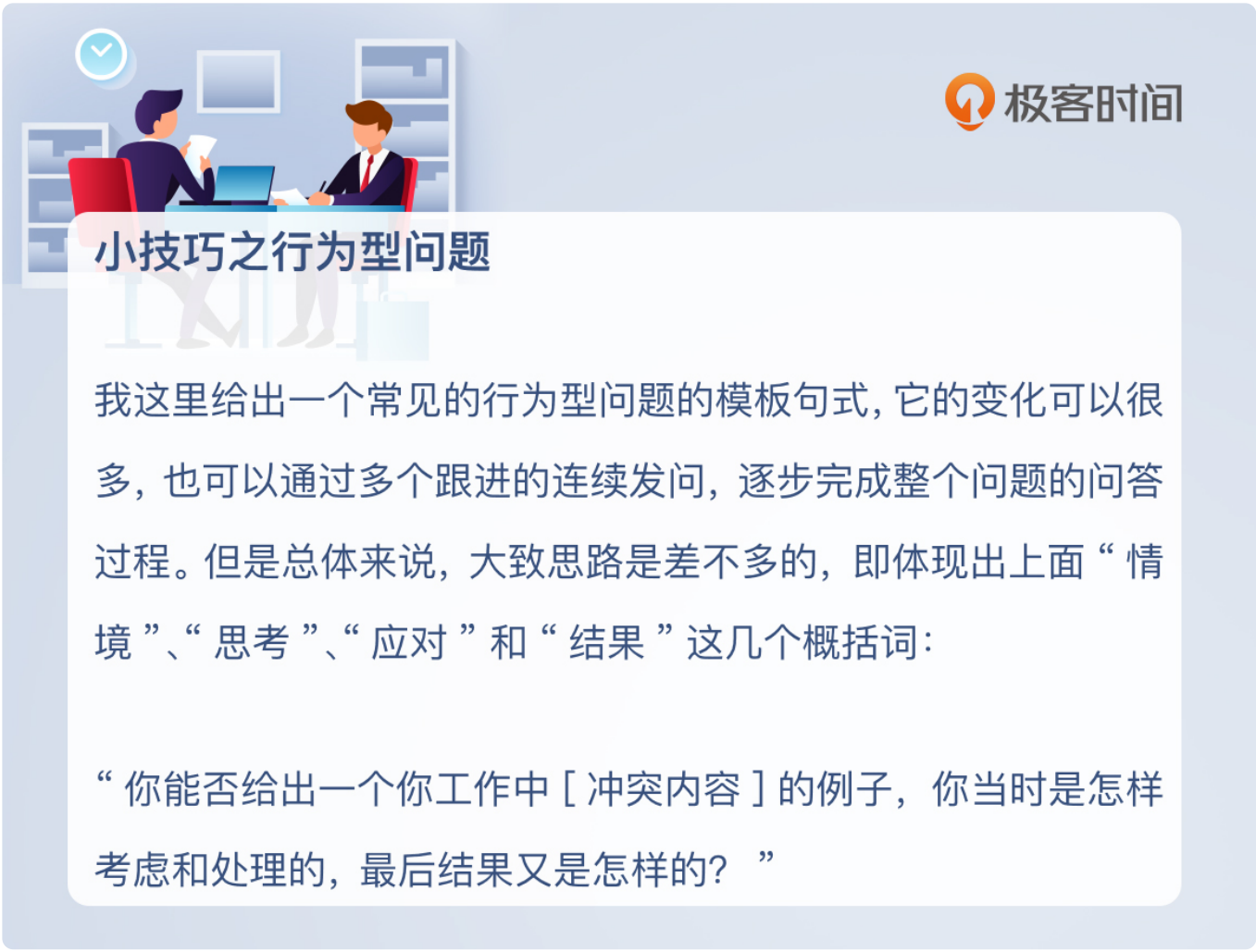
第四，尽量避免太过常见的问题。这和技术问题的设计类似，面试都是可以提前准备的，一些太过常见的问题，可能会变成“背答案”，自然也无法得到真实的考察数据。

第五，事先明确并聚焦考察的数据点，控制问题展开的进度。一般采用行为型问题来进行面试的时候，我们可以逐步展开一个问题，并不断发问，但是需要注意的是，小心不要被候选人“拐跑了”，结果就感觉聊了很多，却没有什么有用的事实数据。

由于这种面试形式，并不像前面我们讨论一个“迷你项目”那样很容易看出其中的主线，因此我们需要很明确到底要着重考察什么，而不是和候选人“随便聊聊过去”。在得到自己想要的信息以后，可以先给讨论的问题收个尾，再开始下一个问题。

最后，我们来回想一下，对于一个行为型问题，**我们希望从候选人那里得到哪些内容？我觉得可以用“情境”、“思考”、“应对”和“结果”这样四个词来概括。**

也就是说，对于当时那种情况，候选人是怎样分析和思考的，于是采取了哪些做法，以及最后的结果如何。当然，在候选人的表达之后，根据内容我们可以从中提炼我们关心的考察数据，那完全是面试官的事了。



总之，**对候选人的性格态度、沟通方法、处事风格等等与团队文化密切相关的要素，我们可以通过行为型问题做比较有针对性的考察，可以说这是对技术层面考察的一个非常必要的补充。**这样的形式并不需要花很多时间，非常精心地准备技术问题，而且讨论非常自由，对于没有工程师技术背景的面试官，更是可以比较顺利地采用，因此这种形式还是比较流行的。

总结与思考

在前面几讲介绍了针对软件工程师候选人的算法和数据结构考察，系统设计考察之后，在这一讲我讲解了一些其它工程技能的考察路径，并对于非常常见的一种非技术范畴的考察方式——怎样问行为型问题做了解读。

我要特别给你强调的是，行为型问题很有价值，但是我们还是要保证整个面试过程中，大部分的时间，还是要放在软件工程相关的技术考察方面，这一点我在第 2 讲的计划制定部分已经说过了。

今天的主要内容就是这些。我想在最后留下一个问题，你能否在留言区和我一起讨论——在你的经历中，都有哪些我们没有介绍到的面试考察形式呢？

好，我是四火，我们下一讲见！

提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 07 | 系统设计能力考察：系统设计内功到底怎么考？

下一篇 答疑课堂02：面试实践篇热点问题解答

精选留言

写留言

> 由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。